SBPDU Frame	Format	Source	Destination
DVMyInfo(*n*)	< *n*, $d_T(n)$, *parent(n)*, *c(n, parent(n))*> where *parent(n)* $\in V \backslash B$	$n \in B$	multicast address of *B*
DVOurInfo(*n, n'*)	< *n, n', d(n, n'), cb(n, n'), c(n, cb(n, n'))*> where $n' \in B$, and *cb(n, n')* $\in V \backslash B$	$n \in B$	$n' \in B$
DVInform(*n, n'*)	< *n, n', d(n, n')*> where $n' \in B \backslash \{n\}$	$n \in B$	$k \in B \backslash \{n\}$ and $k \in N_B'(n)$
DVRecord(*n, n'*)	< *n, n', d(n, n'), FG_A(n, n'), FG_T(n, n'),* *FG_R(n, n')*> where $n' \in B \backslash \{n\}$	$n \in B$	$k \in B \backslash \{n, n'\}$ and $k \in N_B'(n)$

Table 4: Format of DVC SBPDU Frames

FIG. 13 shows an example in which it is not sufficient to calculate the correct distance. Nevertheless, *n* and *j* may obtain an overestimate of the true distance between them by simply adding $d_T(k, n)$ and $d_T(k, j)$. It can be done by requiring *k*, the distant STAR ancestor neighbor, to send the information $d_T(k, j)$ to *n* using a DVInform(*k, j*) frame. If *n* also knows *c(k, m)*, where *m* is the child of *k* on the path to *n*, it can get an even better estimate. Therefore, *m* and *c(k, m)* are sent by *k* in a DVOurInfo*(k, n)* frame. Note that $m \in V \backslash B$. We denote by *cb(k, n)* the child bridge of *k* on a tree path leading from *k* to *n*. Let *dsanc(n)* be used by bridge *n* to keep track of *cb(k, n)*, the child of the *dsan(n)*, and *dcc(n)* to denote the *doubly counted cost* of the link between *cb(k, n)* and *k*. In this case, *dsanc(n)* and *dcc(n)* are respectively set to *cb(k, n)* and *c(k, dsanc(n))*.

Upon receiving a DVMyInfo(*k*) frame <*k, d, n', c*> from tree port *p*, a STAR bridge *self*, with its root path distance $d_T(n)$, processes the frame as in Pseudocode 1. FIG. 14 shows the flow-chart corresponding to Pseudocode 1 (DVMyInfo_Proc). When *p* is a child port, *k* is a descendant of *self* and *self* can calculate the tree path distance between them and inform *k*. There are two cases of interest in the processing of a DVMyInfo frame when *p* is a root port. In Case 1.1, bridges *self* and *k* are siblings with a common old bridge parent. In Case 1.2, *self* and *k* are on different branches but they are not siblings and the distance *d(self, k)* evaluated in this case may be incorrect. It is correct only if the nearest common ancestor of *self* and *k* is the root bridge.

If *n* and *k* do have a STAR common ancestor *n'*, *n* can estimate *d(n, k)* better when it receives the DVOurInfo(*n', n*) and DVInform(*n', k*) frames from *n'*. STAR bridge *self* processes DVOurInfo frame <*n', k, d, m, c*> as in Pseudocode 2. FIG. 15 shows the flow-chart corresponding to Pseudocode 2 (DVOurInfo_Proc).

Each STAR bridge *n* receives at most one DVOurInfo(*n', n*) frame, and this frame must be sent from *dsan(n)*, which is *n'*. In Pseudocode 2, the frame DVOurInfo(*n', k*) is dropped when *self* $\neq k$ because the

recipient of the frame is *self* and there is no other STAR bridge along the path from *self* to *n'*. In Pseudocode 1, a DVOurInfo(*n', n*) frame is sent before sending any DVInform frame. As a DVInform frame will not propagate beyond its distant STAR neighbors and bridges do not reorder frames, we can conclude that *n* receives the DVOurInfo(*n', n*) frame before receiving any DVInform frame sent by *n'*. Therefore, when bridge *n* receives a DVInform(*n', j*) frame <*n', j, d*>, it should have *dsanc(n)* and *dcc(n)* correctly assigned. The pseudocode for bridge *self* to execute when the DVInform(*k, j*) frame <*k, j, d*> is received is shown in Pseudocode 3. FIG. 16 shows the flow-chart corresponding to Pseudocode 3.

A STAR bridge *self* processes a DVInform(*k, j*) frame <*k, j, d*> according to the following different situations:

- Case 3.1: *self* = *j*, that is, *k* is informing *self* the distance between them
- Case 3.2: *self* ≠ *j*, and, either *d(self, k)* or *d(self, j)* is unknown
- Case 3.3: *self* ≠ *j*, *d(self, j)* is estimated
  - Case 3.3a: *parent(self)* = *dsanc(self)*
  - Case 3.3b: *parent(self)* ≠ *dsanc(self)*
- Case 3.4: *self* ≠ *j*, *d(self, k)* is known, and *d(self, j)* is accurate

If *d(self, j)* is accurate, the bridge self does not have to do anything. In Case 3.3 when *FG_A(n, j)* = *0*, it means *d(self, j)* is an estimate and so a better estimate can be obtained. Case 3.3a is the situation where *dsanc(self)* is the nearest common ancestor of *self* and *j*. As a result, *self* can calculate the distance correctly. Since *j* may not be able to do so (see FIG.13), *self* has to inform *j* by a DVInform*(self, j)* frame. When *j* receives that, which is Case 3.1, it can enhance the distance. Case 3.3b is the situation that accurate distance cannot be found but the estimated distance can be improved by taking out the doubly counted cost *dcc(n)*. Since DVInform frames are sent after DVOurInfo and DVMyInfo frames, Case 3.2 occurs only if there is an error.

Ultimately, the first phase must terminate. When it terminates, *n* should have a correct or an overestimate $d_T(n, n')$ for every distant STAR neighbor *n'*. We then proceed to the second phase, that is, each STAR bridge *n* will fill out *DVT(n, k)* if *k* is a direct STAR neighbor of *n*. If *k* is a tree neighbor, *n* should know *c(n, k)* from the topology database, it can initialize the entry *DVT(n, k)* to be *(c(n, k), p(n, k), k, 1, 1, 1)* if *k* is a parent and *(c(n, k), p(n, k), k, 1, 1,-1)* if *k* is a child. If *k* is a crosslink neighbor, there are two cases. If k is not a distant STAR neighbor, *n* doesn't have *DVT(n, k)* yet and so initializes it to be *(c(n, k), p(n, k), k, 1, 0, 0)*. On the other hand, if *k* is also a distant STAR neighbor, *d(n, k)* should, if possible be assigned to be $min(d_T(n, k), c(n, k))$, and other fields accordingly. If the estimated $d_T(n, k)$ is surely correct, that is, if *FG_A(n, k)* = 1, it is trivial. Unfortunately, $d_T(n, k)$ may be incorrect. Since we are not sure whether the direct link *(n, k)* is shorter than the tree path from *n* to *k*, $d_T(n, k)$ won't be replaced in order to avoid selecting a link with a larger distance than its corresponding tree path. The same applies whenever the distance vector is enhanced. When the tree path distance between a pair of STAR bridges is only an estimate, the tree path won't be replaced between them by a

non-tree path. Therefore, when the distance vector becomes stable, $d(n, n') \leq d_T(n, n')$ for all $n, n' \in B$ such that $n \neq n'$.

PROCEDURE: DVMyInfo_Proc(*k, d, n', c, p*), also see Fig. 14.

```
Begin
    If p = p_r(self)                    /* p is the root port, self and k are on different branches */

        If parent(self) = n'                        /* Case 1.1: self and k are siblings */
            d(self, k) := c(self, n') + c ;
            FG_A(self, k) := 1
                                                    /* Case 1.2: self and k are not siblings */
        Else
            d(self, k) := d_r(self)+ d ;            /* overestimated d(self, k) */
            FG_A(self, k) := 0
        Endif
        F(self, k) := p ;
        next(self, k) := k ;
        FG_T(self, k) := 1 ;
        FG_R(self, k) := 0

    Else                                /* p is a child link port, i.e., self is an ancestor of k */
        DVT(self, k) := (d - d_r(self), p, k, 1, 1, -1) ;

        Send DVOurInfo(self, k) frame to k
            <self, k, d(self, k), cb(self, k), c(self, cb(self, k))>
        For each bridge j where F(self, j) = F(self, k)
                                                    /* j and k are from the same child port */
            Send to j DVInform(self, k) frame
                <self, k, d(self, k)>
            Send to k DVInform(self, j) frame
                <self, j, d(self, j)>
    endif
end
```

Pseudocode 1: DVMyInfo_Proc